Massachusetts Institute of Technology
Artificial Intelligence Laboratory

A. I. Memo No. 1186 **October 1989**

Location Recognition Using Stereo Vision

David J. Braunegg

Abstract: A mobile robot must be able to determine its own position in the world. To support truly autonomous navigation, we present a system that builds and maintains its own models of world locations and uses these models to recognize its world position from stereo vision input. The system is designed to be robust with respect to input errors and to respond to a gradually changing world by updating the world location models. We present results from tests of the system that demonstrate its reliability. The model builder and recognition system fit into a planned world modeling system that we describe.

Acknowledgments. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–85–K–0124 and under Army contract number DACA76–85–C–0010 and in part by the Office of Naval Research University Research Initiative Program under Office of Naval Research contract N00014–86–K–0685.

1 Introduction

Much of the current research in navigation planning and world modeling for mobile robots concerns *mission-level autonomy*, i.e., autonomous operation over a limited time frame for the accomplishment of a specific goal. We, however, are interested in the problems of *long-term autonomy*, i.e., autonomous operation of a mobile robot over a period of days, weeks, or months for the continued achievement of a set of goals. For example, there are plans to build a mobile robot that will "live" in our laboratory, continually collecting empty soda cans and returning them to a central repository [Connell].

To achieve navigation over a long lifetime, a mobile robot needs a memory of the world, i.e., a map or "world model." For true autonomy, the robot must be able to navigate in places of which it has no previous knowledge. Thus, we want the robot to build its world model instead of having it supplied *a priori*. Unfortunately, due to the problem of cumulative error, exact metrical models of the world cannot be used [Brooks 1985]. The most promising alternative is a topological map that contains world locations and information about how they are connected [Kuipers] [Chatila and Laumond]. However, due to errors and uncertainty in odometry, we cannot follow such a map exactly. Thus, we need the ability to recognize the locations contained in such a map.

The first part of the problem we are considering, then, is how to *build* models of world locations and *use* them for recognition. The problem becomes more difficult when we realize that the world changes over time. Also, our sensory input is imperfect. Therefore, the second part of the problem is to *maintain* these models over time as the world changes and as we receive new sensory data which is noisy. Thus, the problem we are addressing is to *build*, *use*, and *maintain* models of world locations for recognition to support navigation.

To explore this problem, we are using stereo vision as the sensory input and the 9th floor area in our laboratory as the world. Stereo vision was chosen for sensing because of the spatial resolution of its results. Although sonar is a popular choice for a mobile robot sensor, its poor angular resolution makes it an inappropriate choice for solving the recognition problem. (However, we note that [Drumheller] has used sonar to localize a mobile robot in a known location given an *a priori* model of that location.) The 9th floor of our laboratory was chosen as the test world because we wish to address the problems of indoor navigation in a standard office-type environment.

The problem stated above is difficult for four reasons:

1. Any sensory input is noisy. There will be both errors of omission and commission, i.e., blind spots and hallucinations.

2. The data is sparse. From the Marr-Poggio-Grimson stereo algorithm that we are using, depth information is only available at intensity edges in the images.
3. The world changes over time. Although major features of the world tend to remain stable, the room we are in today does not look exactly as it did yesterday.
4. The robot will have a long lifetime. The system must be sufficiently robust to run for days, weeks, or months without human intervention.

The task, then, is to build a recognition system for world locations. This system as implemented incorporates several pieces: stereo vision, recognition, model building, and model maintenance. Stereo vision is fairly well understood. Progress is being made in the field of recognition. Model building for recognition is fairly new—most recognition systems depend on *a priori* models. We believe that model maintenance is a new area of research for recognition.

To recognize a location (for our testing, one of several connected rooms on the 9th floor of our lab), we first take a series of stereo pair images from a single position in the current room. The stereo vision module (Section 2) then finds salient features of the room and abstracts them into the representation which will be used for recognition (Section 3). Recognition is performed (Section 4) by comparing this representation to room models which were built by the system from similar stereo data obtained previously (Section 5). The results of this recognition are used to update the existing model to reflect the current state of the room and the importance of the features to the recognition (Section 6). This recognition system fits into a larger, planned modeling system (Section 7). Preliminary results of this research are given in the text and Appendix, with a fuller account of the research forthcoming [Braunegg].

2 World Features from Stereo Vision

Any data used to drive a recognition system must fulfill certain requirements. The data must be well-localized, i.e., the locations of the perceived data must have a high enough resolution and correspond closely enough with real-world features to be useful for recognition. The features that the data represent must be distinctive, i.e., must well-characterize the object to be recognized. Finally, the data must be repeatable from different viewing positions. (Many other requirements could also be mentioned, but the ones listed are among those that are essential.)

Stereo vision provides the locations of world features in camera-centered world coordinates. Since these locations have relatively good spatial resolution (compared with sonar data, for example), the stereo data is a good candidate for input to the recognition system. (The error bounds on the stereo data have been investigated by [Matthies and Shafer].)

We use the Marr-Poggio-Grimson stereo algorithm [Marr and Poggio] [Grimson 1981] [Grimson 1985] [Braunegg 1989a] to obtain our stereo data. This stereo algorithm is based on intensity-edge features in the images. Since such features usually correspond to physical edges of visible objects, they characterize the distribution of objects in the visible world. Employing the heuristic that large objects tend not to move and thus help to identify the specific areas in which they reside (e.g., doorways, windows, bookcases), we eliminate short stereo features from the stereo data. Also, since our camera geometry uses horizontal epipolar lines, the localization of the stereo features deteriorates as the features become more horizontal. Thus, we only use long vertical features to represent the indoor spaces through which we navigate (Figures 1 and 2). (Currently, we fit straight line segments to the data [Pavlidis] and eliminate those segments that are shorter than 2 feet tall or have slopes less than 4, i.e., are more than $\approx 25°$ from vertical.)

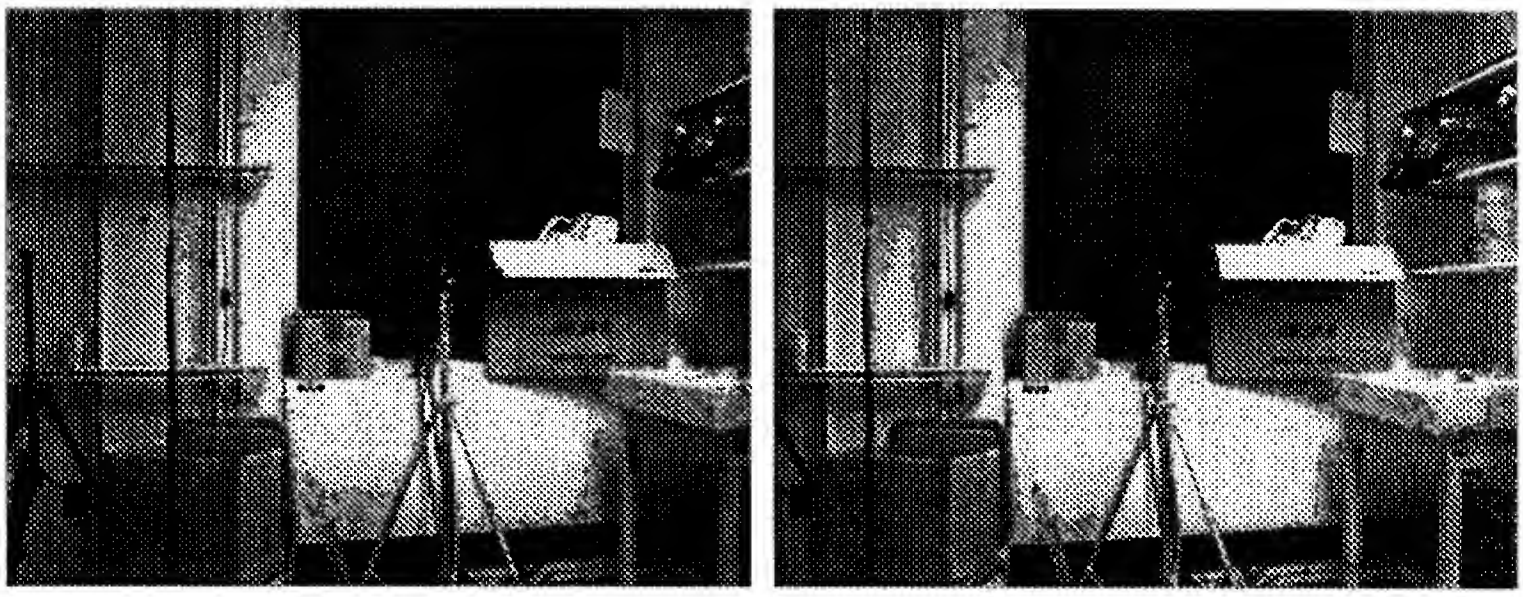


Figure 1. A pair of stereo images used for obtaining room features for Room 914.

To get a full view of the room containing the robot, we rotate the robot through 360°, taking overlapping views of the room [Brooks 1986]. The stereo data from these overlapping views is then "pasted" together for a full representation of the room. Using the camera geometry, the stereo feature coordinates are converted into camera-centered world coordinates and the ceiling and floor features removed. (Using 8.5mm lenses on our Panasonic WVCD-50 cameras,

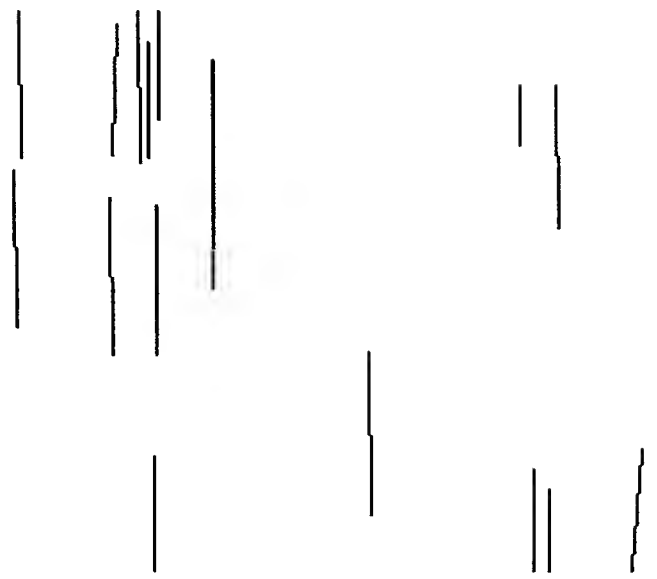


Figure 2. Long, vertical stereo features from the stereo pair in Figure 1.

the field of view is roughly 55°. We take a stereo pair every 20° to obtain sufficient overlap for pasting the data together.)

3 Model/Data Representation

Although it might be tempting to try to create a representation of a room that looks like an architect's floorplan, this is not possible. For a recognition system, the world is not defined as we would like it to be, but instead is defined by what is observable by the sensors that are being used. Thus, in our case, the representation must be related to the matched stereo features obtained from the stereo algorithm.

Other researchers have investigated the use of full 3-D stereo features for map building [Faugeras, Ayache, and Faverjon] [Braunegg 1989b]. However, our task is different in that we wish to recognize rooms rather than navigate through them. To build our room representation for recognition, we project the vertical room features from the stereo algorithm to the groundplane (Figure 3). We have found that this 2-D representation sufficiently characterizes the room for the purpose of recognition. We conjecture that the 2-D representation may also suffice for planning some navigation tasks.

The 2-D groundplane representation of the stereo features has the added benefit of reducing the amount of data that must be handled for each location to be recognized. We further abstract the data by using it to develop an occupancy-grid representation of the current room [Moravec and Elfes]. We impose a grid with 1-foot spacing on the floor of the room and mark each grid square that has a vertical stereo feature falling in it (Figure 4).

Figure 3. Groundplane projection of the vertical stereo features from the full set of stereo pairs for Room 914. The circle shows the camera location and the tick mark in the circle indicates 0° in the camera-based world coordinate system.

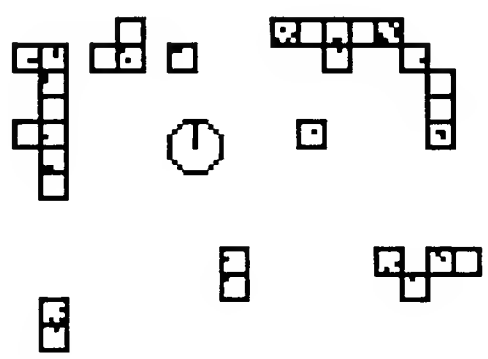


Figure 4. Occupancy grid imposed on the groundplane projection of the vertical stereo features from Figure 3.

The final representation of a room, then, is a set of grid squares that mark the locations of vertical room features as determined by the stereo algorithm (Figure 5). The locations of these squares are described in terms of a camera-centered coordinate system. The orientation of this coordinate system is determined by the orientation of the robot when the first stereo pair is taken. No attempt is made to align the coordinate system axes with the walls of the room.

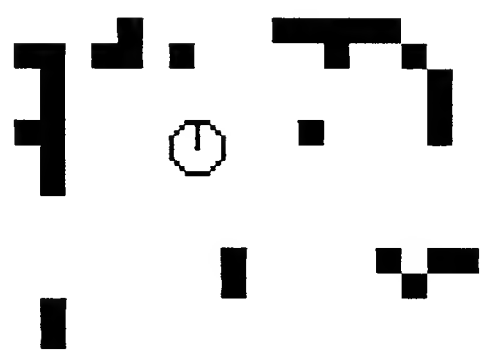


Figure 5. Grid abstraction used to represent Room 914.

4 Recognition

We wish to build the models of locations as the robot explores its world instead of supplying these models *a priori*. The models will therefore be very similar to the data that we obtain. For the first recognition example shown below, the room representation shown in Figure 5 is used as the model. The representation obtained by the robot from another position and orientation in the same room serves as the data in the same recognition example (Figure 6). (The simple model shown here is only the initial model for the room. This room model is updated with the results from each recognition in which it is used (see Sections 5 and 6). Examples of the evolving model are shown in the Appendix.)

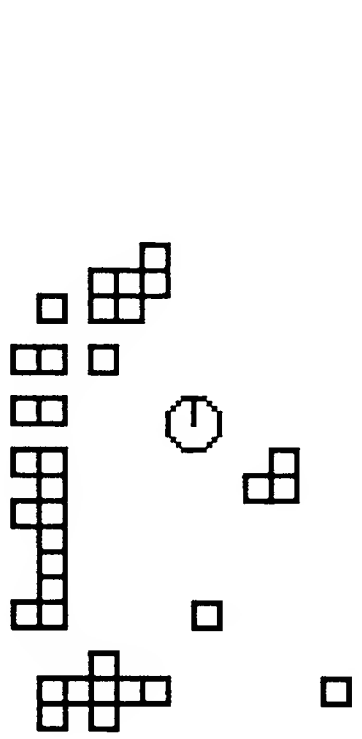


Figure 6. Representation of Room 914 obtained from a different position and orientation of the robot.

To recognize a data set vs. a model, we use a least-squares algorithm to find the best fit of model to data and then evaluate that fit. This fitting process requires a good initial estimate of the transform (2-D translation and rotation) required to match model and data. To obtain these initial estimates, we find transforms that align model and data feature clusters.

The feature clusters used to obtain the initial alignments consist of colinear groups of model and data points. Using all possible pairs of model (data) points to define lines, we find the largest colinear groups of points by selecting the most frequently occurring lines found through a Hough bucketing scheme (Figure 7). We then form pairs of the ten most frequently occurring lines and

keep the pairs that form angles of greater than 20°. For the model and data line pairs whose angles match within a 10° tolerance, we align the line pairs and determine a set of model-to-data transformations based on these alignments. (Other methods of grouping the points of the representations and aligning model and data will need to be added for other environments [Jacobs 1987] [Jacobs 1988], but these linear groups suffice for our indoor scenes.)

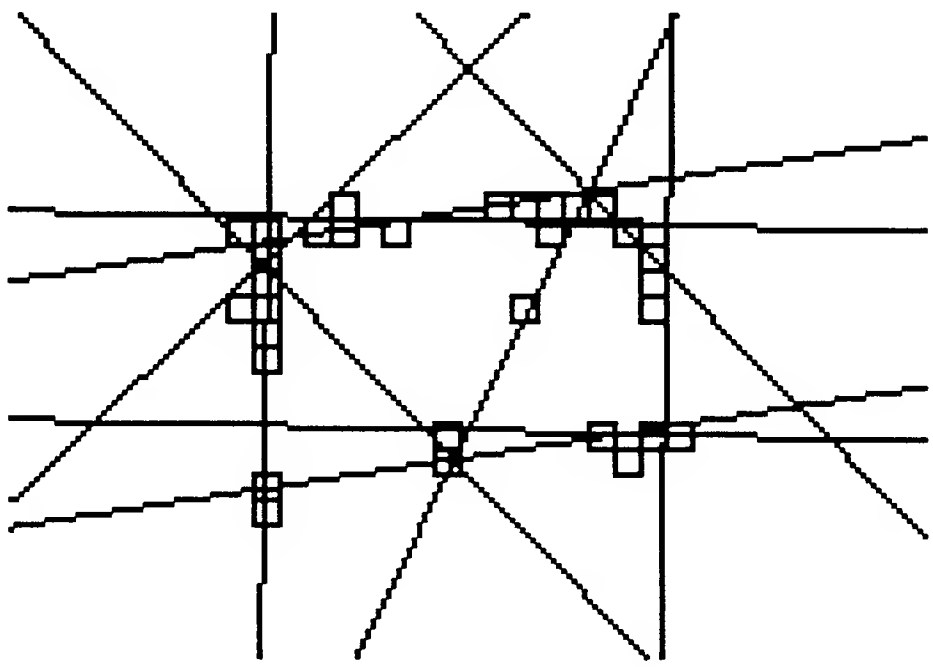


Figure 7. Hough lines for the ten most frequently occurring linear groups of points for the data of Figure 5.

Ranking the transformations by preferring the ones resulting from angles nearest to 90°, we use the top ten transformations as initial guesses for a least-squares model-to-data fit. The least squares minimizes the error between the locations of the model and data points by varying the translation (x, y) and rotation θ of the model. This process is similar to the one described by [Lowe]. However, given an alignment, we find all possible model-data point matches before performing the least-squares optimization instead of adding the pairs as we incrementally refine the transformation. Initially we tried the incremental refinement approach, but found that one bad match could affect the rotation component of the transformation enough to generate a completely wrong final result. A different incremental approach was taken by [Ayache and Faugeras] in their recognition system HYPER, which used a Kalman filter to refine the initial transformation guesses. When we applied this approach to our point data, however, the same problem of one bad match corrupting the final solution still occurred.

The quality of the final recognition is determined by the number of model points that have been matched and the transform variance. The transform variance is the sample variance as determined by summing the squared distances between each model point and its closest data point, then dividing by

the number of model points minus one [Walpole and Myers]. The best recognition obtained from the set of initial guesses is the one that matches the most model points. If more than one recognition matches this same highest number of model points, then the one with the lowest transform variance is chosen (Figure 8).

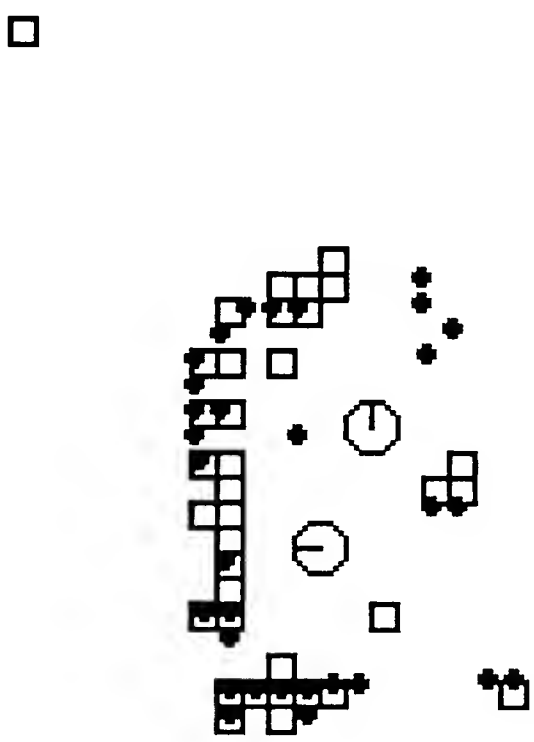


Figure 8. Final recognition using the model from Figure 5 (shown as solid dots) and the data from Figure 6 (shown as open squares).

After finding the best match of model to data, we still need to determine if the match is correct, i.e., if the model and data actually correspond to the same location. Since we need to compare the recognitions of several models to a single data set, we base our evaluation of the recognitions on the data so that the evaluations are comparable. We take as the correct recognition, then, the model that matches the highest number of data points above a threshold. Since it is possible that the data does not correspond to a model currently in our database, the threshold is used so that we can declare no recognition in cases where the match is poor.

Matching the wrong model to the data (a false positive) is a serious error since the mobile robot uses this information to verify its location in its world map. Deciding that the current data does not match one of the models in the database when the mobile robot is actually in one of the locations represented by a model (a false negative) is less serious. With no recognition (with respect to an existing model), the current data is added to the database as a new room model. Over the lifetime of the robot, we will be able to merge this new model into the existing model for this room (see Section 7). With 11 data sets

obtained from one room (Room 914) and 6 data sets from another (Room 913) we ran a series of recognition tests. Using each data set in turn as data with the other 16 as models, we ran the recognition algorithm. A recognition threshold of 70% matched data points yielded 10% false negatives with no false positives. The two rooms 913 and 914 were explicitly chosen because they have similar dimensions, thus intentionally making the recognition difficult. Tests against data from a third room that is a different size yielded no recognition errors. In all cases, recognitions that were accepted also determined the correct model-data transformation.

Once a recognition has been established, we have verified our location with respect to the world model. This serves to confirm that the mobile robot is following the map correctly. Also, since there are error and uncertainties associated with the odometry of a mobile robot, we would like to refine the current estimate of our position. The transformation that is determined between the model and the data via the recognition also serves to give us the current position of the mobile robot with respect to the model. We can then reinitialize the odometry. Also, assuming that information about the positions of important items in the world (e.g., doors—see Section 7) has been associated with the room models, the model-data transformation serves to tell us where these items may be found with respect to our current position.

5 Model Building

In the recognition above, we used the data obtained from one 360° view of a room for our model. This suffices for an initial model, but a more reliable model can be constructed by combining later views of the same room. Once a model-data recognition has been established, the data is transformed to the same position and orientation as the model and the two combined into a new model. A weight is associated with each point in the model and the weight is increased for those model points that are overlapped by data points (Figure 9). If the current data corresponds to no model in the database, that set of data is entered into the database as the model of a new location.

Combining the new data with the current model serves two purposes. First, new features that appear in the data are added to the model. This is important since some room features may be occluded from certain viewpoints. Second, the features that were used for recognition (and thus had a model-data overlap) become more important in the recognition process. This is accomplished by using a weighted least-squares algorithm to emphasize the

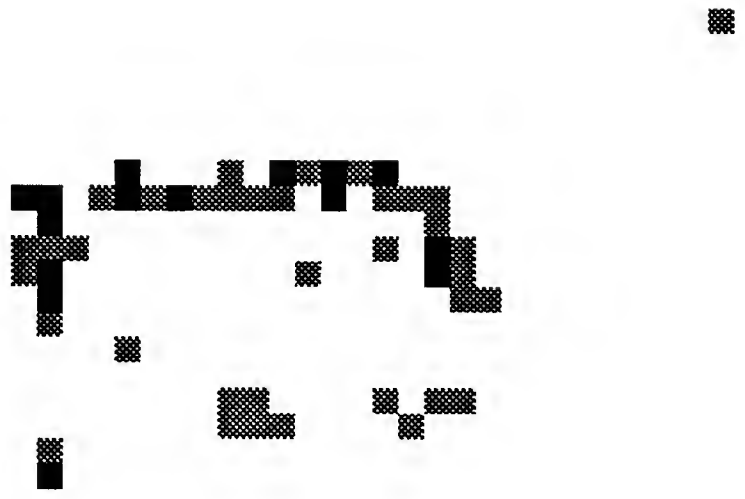


Figure 9. Updated model of Room 914 based on the recognition shown in Figure 8. The darker squares correspond to more heavily weighted model points.

matching of heavily weighted model points. Other advantages to a weighted model will be discussed in the next section.

6 Model Maintenance

By combining the model and data sets of a recognition into a weighted model, we build the model up over time. The way in which these weights are determined obviously affects the appearance of the new model. Therefore, we are currently investigating different schemes for incrementing the model-point weights. Our current scheme weights the initial model points by one and increments these weights by one each time a data point overlaps them. A different possibility is to weight the currently visible points more heavily than the points already extant in the model. The justification is that the models represent our *memory* of the world while the current data represents the world as it currently exists and thus is more certain. A further refinement is to *time*-weight the model points so that the longer ago we saw a model feature, the less importance we give it for recognition. This weighting can be based on a count of the number of times a model is used or on the actual clock time since the model was last used. In all of these weighting schemes, there is a low-weight threshold. When the lowest weights of a model become sufficiently small, they are eliminated from the model. This step is obvious once we realize that the weights reflect our confidence in the existence of the various model features.

All of the model-weight update schemes have a common theme: the more often we see something, the more confident we are of its existence. This is important for three reasons. First, the world changes. By adding newly observed world features into the models, we can incorporate features belonging to objects that are new or are in new locations. By removing those model features that have low weights, we remove from the models those world features that no longer exist or that are no longer in their original locations. (The feature weights will be low because, since the features are no longer visible, they will not exist in the data sets that are being used to update the model weights.)

Second, no stereo vision algorithm (or any other sensing scheme) is perfect. There will be both errors of omission and commission, i.e., blind spots and hallucinations. The blind spots will be filled in by later data that is incorporated into the models. The hallucinations will slowly disappear since they will not be seen again and their weights not reinforced.

Third, the locations of world features determined by the stereo algorithm (or, again, any other sensing scheme) are not perfect. When a world feature is seen several times and its location entered into the model, the average perceived location gradually increases in weight. As the data points are added over time, the marked model points approximate a Gaussian distribution about the true location of the feature. Thus, the model updates also serve to refine the locations of the world features in the model. This approach avoids the additional computation needed to model the uncertainty of the sensor data [Durrant-Whyte] [Matthies and Shafer].

7 Location Recognition and the World Model

The location (room) models described above are part of a larger world modeling system that is planned. The mobile robot's world will be represented by a topological map similar to that described by [Kuipers]. The world will be represented by locations and information about how they are connected. Rather than being supplied *a priori*, this world model will be built by the robot as it explores its environment. Because the robot has long-term autonomy, existing for days, weeks, even months at a time in its environment, the amount of time incurred to build the initial world model is amortized over the lifetime of the robot. Because the robot builds and maintains its own world model, it can react to a (gradually) changing world in a way that a robot depending on a precomputed map cannot.

For each location in the world, a model as described previously will be built. Since the world locations are discovered through movement (exploration) in the environment, the connections between the locations will be traversed and the location models can be annotated with their positions. Other sensing and robot tasks will assign importance to places and features in the environment and these, too, can be annotated on the world model.

When a room is entered, the robot will check the database of known locations to see if it is in a previously encountered location. If so, the current data will be incorporated into the current model as described above. If this is the first time the room has been entered, the robot can wander about the room and take data from several viewpoints. Knowing that these viewpoints all come from the same world location, they can be combined into the initial model for that location. A background process will continually compare the location models in the database to search for multiple models that have been created for the same location due to false negatives from the recognition system. (As more data is added to the models, the models can be recognized as actually representing the same location.) Multiple models for the same locations will then be combined in the same way that model and data are, as described previously. The end effect will be that duplicate sections of the world will be compacted into a more concise representation.

By following the world model as it travels through its environment, the robot always has an estimate of its current location. Given this current estimate, only the model for the estimated location and neighboring locations need be compared to the current data to verify the current location. Thus, although the database of locations might be quite large, only a small fraction of the models need be compared to the data at any one time. If, for some reason, the robot becomes completely lost and has no current estimate of its location in the world model, the currently observed data can be compared to all of the location models in the database. This may take some time if the database is large, but the robot can do no useful work until it finds itself in any case!

The mobility of the sensing platform (a mobile robot) can be used in two different ways. If the observed data ambiguously matches two different location models, the robot can be moved to a location where it can explicitly look for features that are not common between the models. If this fails, the robot can explore the local area, build up a second map of the world, and compare this map to the current world model to determine its location.

Thus, the world location recognition system as implemented fits nicely into our larger world modeling scheme. Just as the models of the individual

world locations evolve over time, so does the world model itself. The complete system is well-suited to a long-term autonomous mobile robot.

8 Conclusions

We have presented a working system that builds, maintains, and uses models of world locations for recognition to support navigation. This system uses stereo vision to obtain information about the world for use as input to the model builder and the recognizer. In addition, we have described how this recognition system will fit into a larger world modeling scheme.

This work is interesting for several reasons. The model builder and recognition system uses sparse data from the real world as input. We accept the fact that any input data is noisy and explicitly provide a method for handling these data errors. We acknowledge that the world is not static and therefore we provide a way for our location models to change over time as the perceived world changes. We provide for long-term autonomy in our mobile robot by rejecting *a priori* models in favor of models built by the robot itself, thus allowing the robot to explore areas of the world. And finally, we not only build the models that are used by the recognition system, we also maintain them over time in the presence of sensing errors and a changing world.

The recognition system has been tested on over 300 model-data pairs from actual scenes with a 10% false negative rate and a 0% false positive rate for the initial unity-weight models. The error rates are even lower when using weighted models that have been built up over time. The error rates demonstrated are low enough to permit the inclusion of this world location recognition system in the larger world modeling scheme that we have outlined.

9 Further Work

We are continuing with the testing of the system as described as well as extending it. Different update schemes for the models weights need to be tested. Groupings other than linear clusters of features should be examined in order to obtain better initial alignment guesses. The issue of separating different locations (rooms) needs to be addressed. We are currently concentrating on offline examinations of the observed data to separate rooms based on geometrical properties of the data. Other sensing methods, such as sonar, could also

complement the system by providing information about the passage of the robot through restrictions, such as doors and hallways, that separate rooms.

Further out on the horizon, non-indoor environments need to be considered. Our reliance on simple sparse, rather than extended dense, features will help us extend this work to outdoor environments where, for example, scattered trees might provide the world features that identify a particular field.

Also for the future is an examination of other sensors to provide the data for recognition. From this work, we have learned that the sensors must provide repeatable measurements of distinct features in the world with reasonable resolution. Laser ranging systems are certainly a possibility and narrow-beam active sonar might also hold some promise.

10 References

Ayache, N. and O. D. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–8** (1), January 1986, 44–54.

Braunegg, David J., "Stereo Feature Matching in Disparity Space," MIT AI Lab, AI Memo 1184, September 1989a.

Braunegg, David J., "An Alternative to Using the 3-D Delaunay Tessellation for Representing Freespace," MIT AI Lab, AI Memo 1185, September 1989b.

Braunegg, David J., "Recognizing World Locations with Stereo Vision," Doctor of Philosophy thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, expected June 1990.

Brooks, R. A., "Visual Map Making for a Mobile Robot," *IEEE International Conference on Robotics and Automation*, St. Louis, MO, March, 1985, 824–829.

Brooks, R. A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, **RA-2** (1), April 1986.

Chatila, R., and J.-P. Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots," *IEEE International Conference on Robotics and Automation*, St. Louis, MO, March, 1985, 138–145.

Connell, J. H., "A Colony Architecture for an Artificial Creature," MIT AI Lab, Technical Report AI-TR-1151, 1989.

Drumheller, M., "Mobile Robot Localization Using Sonar," MIT AI Lab, AI Memo 826, January 1985.

Durrant-Whyte, H. F., "Sensor Models and Multisensor Integration," *The International Journal of Robotics Research*, **7** (6), December 1988, 97–113.

Faugeras, O. D., N. Ayache, and B. Faverjon, "Building Visual Maps by Combining Noisy Stereo Measurements," *IEEE International Conference on Robotics and Automation*, San Francisco, CA, April, 1986, 1433–1438.

Grimson, W. E. L., *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA, 1981.

Grimson, W. E. L., "Computational Experiments with a Feature Based Stereo Algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–7** (1), January 1985, 17–34.

Jacobs, D. W., "Groper: A Grouping Based Recognition System for Two Dimensions," *IEEE Computer Society Workshop on Computer Vision*, Miami Beach, FL, November 30–December 2, 1987.

Jacobs, D. W., "The Use of Grouping in Visual Object Recognition," MIT AI Lab, Technical Report AI-TR-1023, October 1988.

Kuipers, B. J., "Representing Knowledge of Large-Scale Space," MIT AI Lab, Technical Report AI-TR-418, July 1977.

Lowe, D. G., "Three-Dimensional Object Recognition from Single Two-Dimensional Images," *Artificial Intelligence*, **31**, 1987, 355–395.

Marr, D. and T. Poggio, "A Theory of Human Stereo Vision," *Proceedings of the Royal Society of London*, **B** (204), 1979, 301–328.

Matthies, L. and S. A. Shafer, "Error Modelling in Stereo Navigation," Computer Science Department, Carnegie-Mellon University, CMU-CS-86-140, 1986.

Moravec, H. P., and A. Elfes, "High Resolution Maps from Wide Angle Sonar," *IEEE International Conference on Robotics and Automation*, St. Louis, MO, March, 1985, 116–121.

Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, MD, 1982, 287–290.

Walpole, R. E. and R. H. Myers, *Probability and Statistics for Engineers and Scientists*, Macmillan Publishing Company, New York, 1978.

Appendix

We present some of the data obtained from the two rooms mentioned in this paper along with the weighted models built from this data. Figure 10 shows some of the data sets taken from Room 914. The first data set was used as the initial model and the second as the initial data. After recognition, an updated weighted model was formed and used in the recognition algorithm with the third data set, again producing an updated weighted model. This process was repeated for all the data sets shown. Figure 11 shows the weighted model as it evolves over the course of the various updates. Note how the extraneous data point from the second data set fades away. The points inside the upper right-hand corner of the model correspond to a wire that was hanging from the ceiling when the first three data sets were taken. Note that these points also fade away since the wire was not present in the later data sets. Figure 12 shows the final weighted model for Room 914 before and after the low-weight points are removed. Note how the location of the bottom-right model point is being refined and also that the extraneous data point from the second data set has been removed.

Figures 13, 14, and 15 show, respectively, data sets for Room 913, the evolving model for Room 913, and the final Room 913 weighted model.

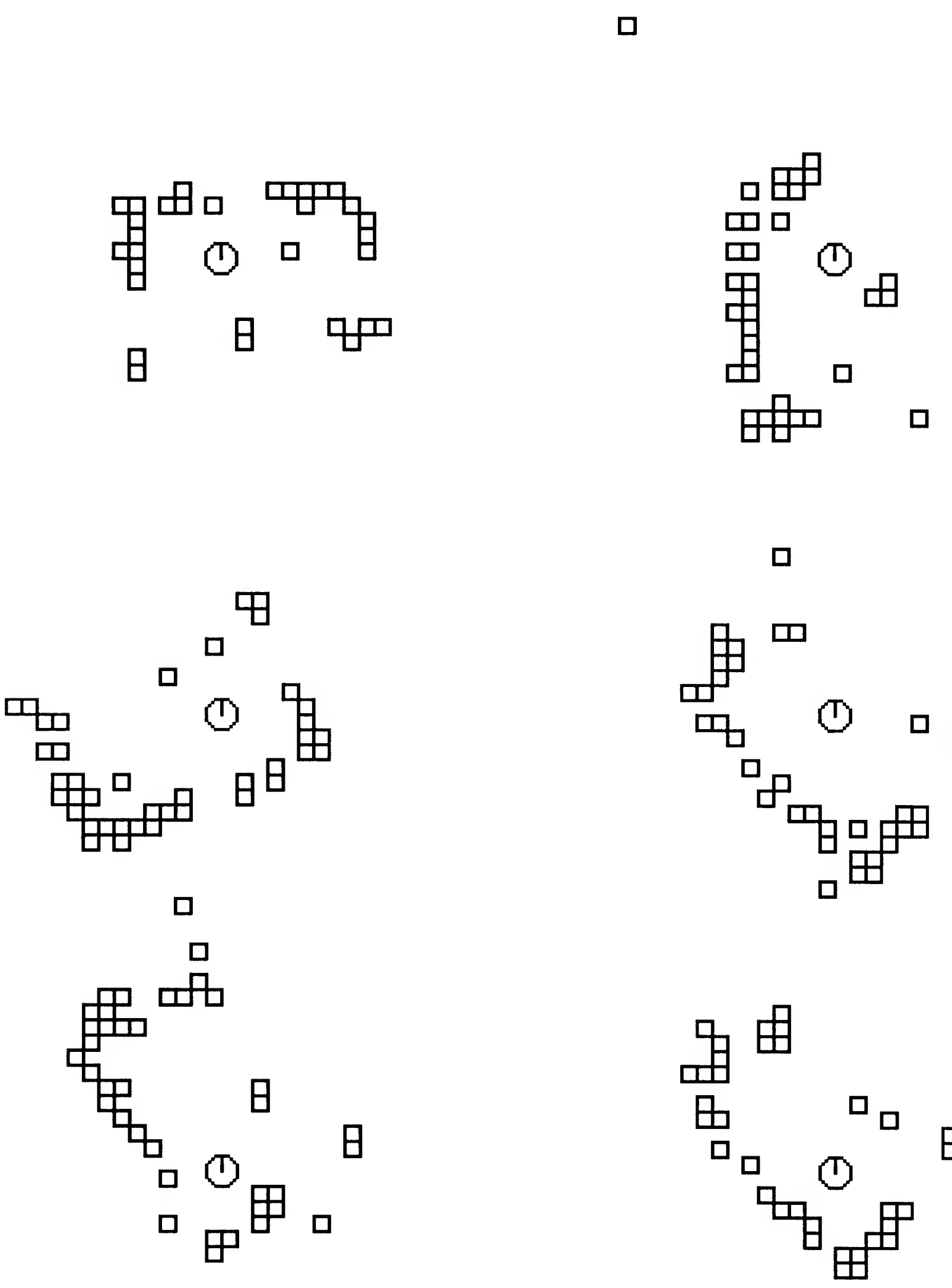
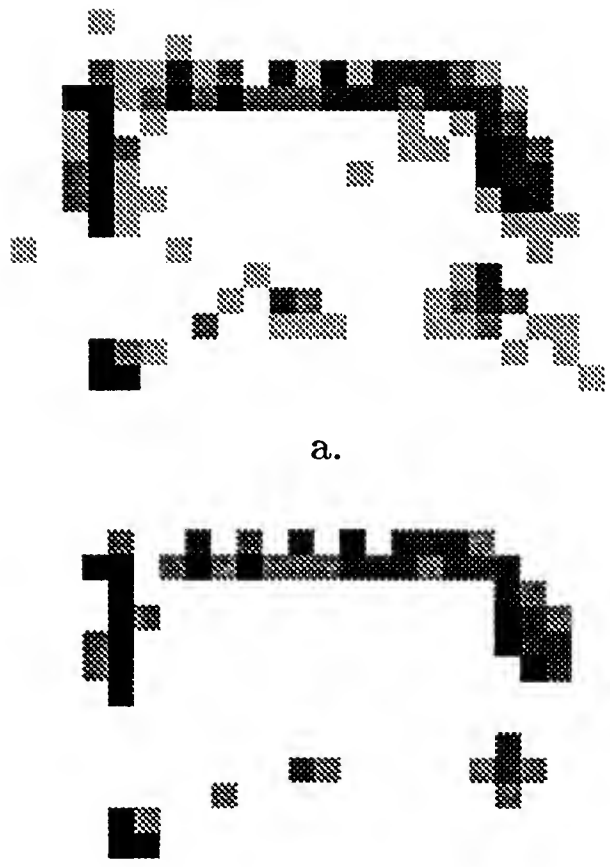


Figure 10. Data sets for Room 914.

Figure 11. The weighted model for Room 914 as it evolves over the course of several recognitions. The darker squares correspond to more heavily weighted model points.



a.

b.

Figure 12. The final weighted model for Room 914. a. Before low-weight point elimination. b. After low-weight point elimination.

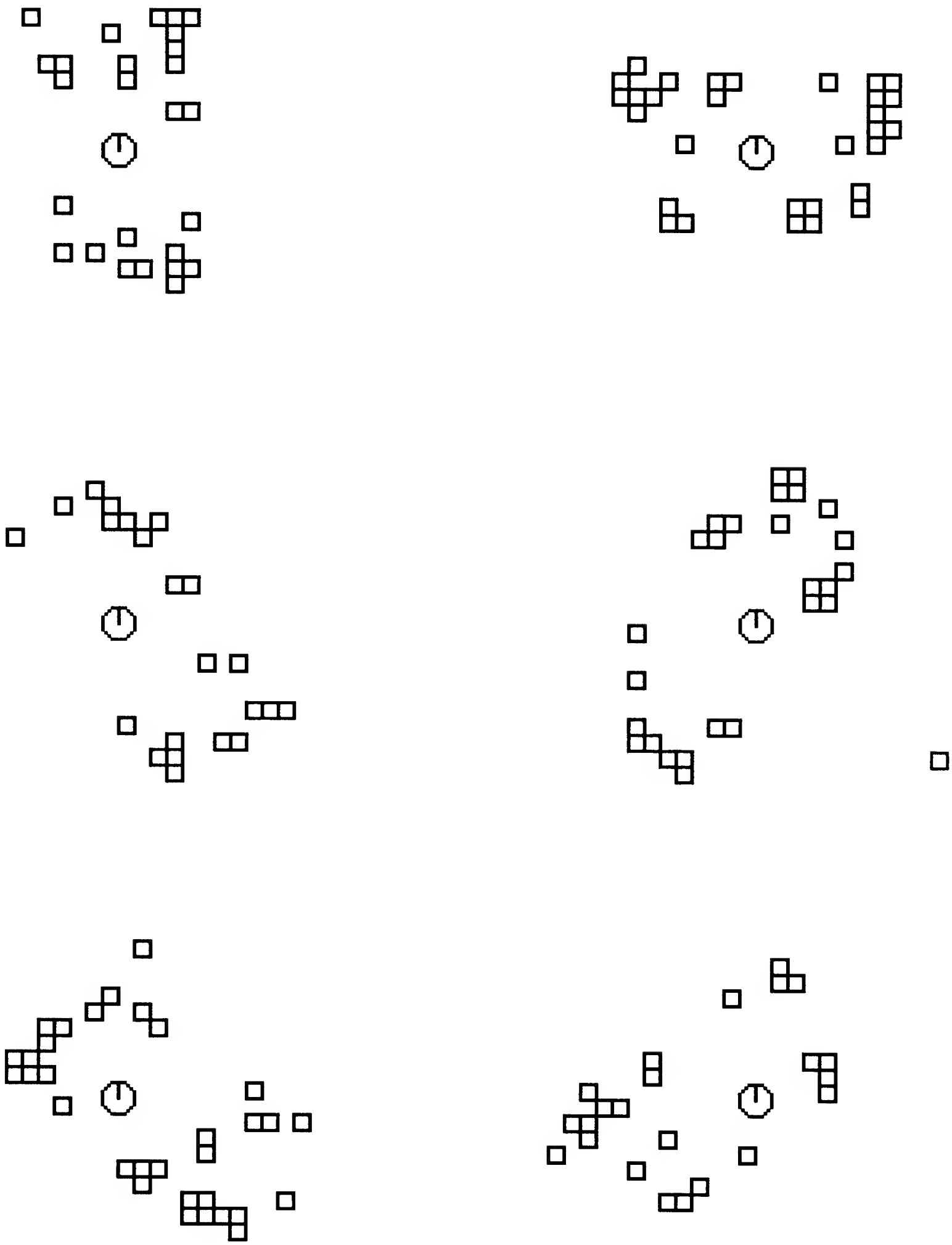
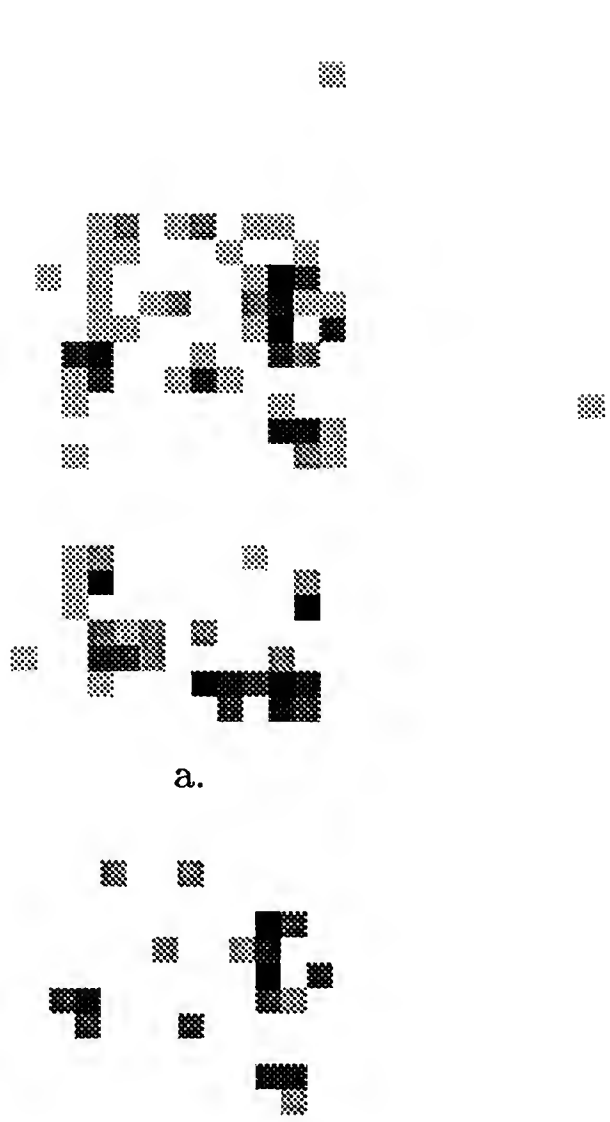
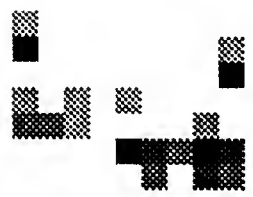


Figure 13. Data sets for Room 913.

Figure 14. The weighted model for Room 913 as it evolves over the course of several recognitions. The darker squares correspond to more heavily weighted model points.



a.



b.

Figure 15. The final weighted model for Room 913. A. Before low-weight point elimination. B. After low-weight point elimination.